A bilingual MOOC that teaches youngsters how to program: Analysis and reflections on one year of experiences

A bilingual MOOC that teaches youngsters how to program: Analysis and reflections on one year of experiences

Inés Friss de Kereki, J. Víctor Paulós
Universidad ORT Uruguay
Facultad de Ingeniería
Montevideo, Uruguay
kereki_i@ort.edu.uy, paulos@ort.edu.uy

Areti Manataki
University of Edinburgh
School of Informatics
Edinburgh, United Kingdom
A.Manataki@ed.ac.uk

***Abstract*— Critical thinking and problem solving are fundamental skills to function successfully particularly in today's world. When programming, these skills are promoted, developed and deployed. In this context, Universidad ORT Uruguay and The University of Edinburgh co-created in 2015 a MOOC (Massive Open Online Course) that teaches young teenagers how to program. The course was implemented simultaneously in 2 versions: in Spanish, called "¡A Programar!" and in English, called "Code Yourself!", which are available on the Coursera platform. Since its launch in March 2015, more than 139,000 people from 197 countries have registered. Initially it was offered in a "fixed session"; while currently it is offered in an "auto-cohort" mode. In both cases, students' surveys indicate that the course has met or exceeded expectations (values above 93%). In this paper, we detail the characteristics of the MOOC, and we analyze and compare the modes and results.**

Keywords—MOOC, programming, computational thinking, Scratch

I. INTRODUCCION

El mundo ha cambiado tan fundamentalmente en las últimas décadas que los roles de enseñar y educar también han cambiado [1]. Para desempeñarse en la sociedad, todo ciudadano en el siglo 21 debe al menos entender los principios de las ciencias de la computación [2]. Las habilidades adquiridas al codificar ayudan a entender la sociedad actual y fomentan las competencias del siglo 21 [3]. Los MOOCs (cursos masivos abiertos en línea, por sus siglas en inglés MOOC: "Massive Open Online Course") son una poderosa herramienta para popularizar el acceso a la educación y democratizar el conocimiento. Dada la importancia cada vez mayor de conocer y adquirir dichas habilidades relativas a ciencias de la computación y también con los objetivos de mayor inclusión e igualdad de oportunidades, la Universidad ORT Uruguay y The University of Edinburgh diseñaron en forma conjunta un MOOC bilingüe: "¡A Programar!" [4] en español y "Code Yourself!" [5] en inglés, dirigido a jóvenes.

En este trabajo se presentan las características y se analizan comparativamente las dos modalidades ofrecidas. En la Sección II se presentan conceptos relacionados al pensamiento computacional así como la inclusión de la programación y ciencias de la computación en los planes de estudio actuales para los jóvenes. En la Sección III se analizan posibles recursos para aprender a programar. En la Sección IV se introducen nociones sobre MOOC incluyéndose opciones para su uso, recomendaciones para su diseño y plataformas disponibles; también se analizan varios MOOCs en el contexto de las ciencias de la computación. En la siguiente sección (sección V) se describen "¡A Programar!" [4] y "Code Yourself!" [5]. En la Sección VI se presentan y analizan las dos modalidades en las cuales fue ofrecido este MOOC. En la Sección VII se incluyen resultados y finalmente se brindan conclusiones y trabajos futuros en la Sección VIII.

II. PENSAMIENTO COMPUTACIONAL

El pensamiento crítico y las habilidades de resolución de problemas son competencias esenciales que los estudiantes deben aprender para el éxito en el mundo actual [6]. En la última década, la educación en tecnologías de la información y de la comunicación (TIC) ha pasado de enfocarse en ser una herramienta a orientarse hacia la comprensión de los conceptos que sustentan las tecnologías digitales [7]. Los trabajos actuales requieren resolver problemas no estructurados y comunicación, por lo cual la preparación de los jóvenes para el mundo laboral debe adaptarse [8].

El pensamiento computacional (PC) es una habilidad fundamental para todos. Incluye resolución de problemas, diseñar sistemas y entender el comportamiento humano. Usa abstracción y descomposición [9]. Es un proceso de resolución de problemas que incluye representar datos a través de abstracciones como modelos o simulaciones, identificar, analizar e implementar posibles soluciones para alcanzar la más efectiva y eficiente combinación de pasos y recursos, generalizar y transferir este proceso de resolución a una amplia variedad de problemas [10]. El PC se infiltra en todas las áreas, tanto de ciencias como de humanidades y está haciendo que cambie la forma en que pensamos [11]. En una sociedad cada vez más basada en la información, el PC se convierte en una habilidad esencial [12].

"Codificar no es un conjunto de habilidades técnicas sino un nuevo tipo de alfabetización y expresión personal, valiosa

El desarrollo de "¡A Programar!" fue patrocinado por "Santander Universidades".

978-1-5090-1633-4/16/$31.00 ©2016 IEEE

para cada uno", es una "nueva forma para las personas de organizar, expresar y compartir sus ideas" [13]. Estudiar computación involucra mucho más que aprender a codificar. Introduce formas de procesar información y representarla, trabajar sistemáticamente para identificar y eliminar errores, y brinda una manera de dividir los problemas y resolverlos [14]. Programar es, no sólo una habilidad fundamental de las ciencias de la computación y herramienta clave para apoyar las tareas cognitivas implicadas en el pensamiento computacional, sino también una demostración de las competencias computacionales [15]. Las competencias empleadas al codificar son la parte más visible de una forma de pensar. Se trata del desarrollo de un pensamiento específico: el pensamiento computacional [16].

Hace varios años, la educación en computación comenzaba en las universidades [17]. Actualmente, la programación y las ciencias de la computación han sido o están siendo incluidas en los programas escolares [18]. Algunos ejemplos los ofrece European Schoolnet [3], donde se reportan 16 países que integran la codificación al currículo nacional, regional o local (Inglaterra, España y Francia entre otros) [3]. En particular, en el currículo nacional de Inglaterra se indica como metas que todos los estudiantes puedan entender y aplicar los principios fundamentales de la ciencia de la computación, y analizar problemas en términos computacionales, entre otros objetivos [19]. En los estándares para la educación en ciencias de la computación desde nivel elemental a secundario en Estados Unidos se indica introducir los conceptos fundamentales de estas ciencias, a todos los estudiantes [2].

III. Recursos para aprender a Programar

Actualmente hay varias iniciativas y recursos disponibles para los jóvenes para aprender a programar. Code.org [20], por ejemplo, provee variados videos y lecciones breves para crear programas encastrando bloques. CodeAcademy [21] ofrece una plataforma interactiva donde se pueden aprender diversos lenguajes, como por ejemplo Python o Java. También hay lenguajes de programación especialmente diseñados para jóvenes. Entre ellos, podemos citar Scratch [22], Alice [23] y AppInventor [24]. Este tipo de lenguajes son particularmente beneficiosos porque requieren menos escritura (se arrastran bloques), son visuales y más atractivos que los lenguajes basados en texto [25].

En particular, Scratch es un lenguaje de programación desarrollado por MIT Media Lab. Permite crear animaciones, juegos e historias. Está diseñado para ser divertido, educativo y fácil de aprender. Está orientado a niños y jóvenes entre 8 y 16 años [22].

Programar en Scratch y compartir los proyectos permite que los estudiantes aprendan importantes conceptos matemáticos y computacionales, así como pensar creativamente, razonar sistemáticamente y trabajar en forma colaborativa, todas habilidades esenciales para el siglo 21 [26]. A través del uso de Scratch se aplican conceptos de computación tales como secuencia, eventos e iteraciones, así como prácticas de computación tales como abstracción y modularización. Todos estos elementos están relacionados al pensamiento computacional [27]. Por ejemplo, el uso de abstracción implica decidir escenas y personajes para incluir o no en las historias y juegos creados en Scratch. Es un ejemplo de aplicación de pensamiento computacional [8].

IV. Acerca de los MOOCs

La educación a distancia no es nueva. Lo que es nuevo es su escala [28]. Un MOOC es un curso en línea, masivo y abierto, ofrecido a través de Internet y disponible de forma gratuita a un número muy grande de personas [29][30]. Permite a los estudiantes acceder a contenido académico de alta calidad [31]. Los MOOCs asisten al desarrollo de las habilidades de los individuos para participar en la economía digital desarrollando habilidades de colaboración con otras personas en línea y desarrollando artefactos digitales [32].

Los MOOCs son una expansión de la educación en línea y de la educación a distancia, que han tenido rápido desarrollo y crecimiento [33]. Algunas razones son el ancho de banda suficiente para la visualización de videos, las mejoras en las tecnologías web y los avances en las plataformas [30]. Los MOOCs tienen el potencial de proveer educación en escala global [34]. También se debe considerar que los MOOCs pueden ser además medios efectivos para educar estudiantes cuando no son posibles o no están disponibles formas presenciales [34]. Permiten asimismo una variedad de estilos de clases. Button propone incluir las fortalezas de un MOOC en otras formas de enseñar [35]. Por ejemplo, usarlo en clases invertidas (donde el estudiante debe realizar ciertas actividades previas a la clase aprovechando los recursos de alta calidad de un buen MOOC), permitir estudios independientes, y, o, mejorar clases tradicionales con otros materiales [30].

Algunas recomendaciones para el diseño de MOOCs brinda Button [35]. En particular, recomienda que la duración de un MOOC sea de unas cuatro semanas, los videos deben ser de pocos minutos, y preferir que los profesores proyecten un aire más "informal" sentados frente a un escritorio que de pie en un podio [35]. Alario et al [36] también señalan entre 5 y 10 minutos como la duración recomendada de los videos y combinar diversos tipos de recursos, tales como animaciones, dibujos o capturas de pantalla. Guo et al [37] señalan que los videos en los cuales los instructores hablan más rápido y con gran entusiasmo son mucho más motivadores. Para lograrlo, sugieren entrenarlos para mostrar su entusiasmo [37]. De acuerdo con Adamopoulos [38], cuanto más satisfecho esté el estudiante con el profesor, los materiales y las tareas a realizar, más probablemente termine el curso en forma exitosa. Además, la evaluación por pares tiene efecto positivo en la compleción del curso.

Hay múltiples plataformas para ofrecer MOOCs. Entre las más reconocidas podemos citar Coursera [39], edX [40] y Udacity [41]. En español está disponible Miríada [42]. Coursera ofrece más de 1600 cursos [39], edX más de 900 [40], Udacity más de 140 cursos [41] y Miríada más de 375 cursos [42]. Respecto a cantidad de alumnos, Coursera dispone de más de 15.000.000 de estudiantes, edX refiere más de 5.000.000 de estudiantes, Udacity indica 4.000.000 de usuarios [43] y Miríada informa de unos 2.000.000 de alumnos [42].

Hay variedad de MOOCs para aprender los conceptos básicos de las ciencias de la computación. La mayoría de ellos

utilizan lenguajes de programación de uso profesional, como Java o Python. Por ejemplo, "CS50x: Introducción a las ciencias de la computación" [44], ofrecido en inglés en la plataforma edX. Dura 12 semanas, enseña como pensar algorítmicamente y resolver problemas en forma eficiente. Algunos de los tópicos cubiertos son: abstracción, algoritmo, estructuras de datos, encapsulación y seguridad. Utiliza múltiples lenguajes: C, PHP, JavaScript y Scratch, entre otros. Es un curso muy estimulante pero puede ser muy difícil de seguir para jóvenes y principiantes, porque el nivel de dificultad rápidamente crece en el curso. "Introducción a las ciencias de la computación" [45], disponible en Udacity, utiliza Python como lenguaje y dura 3 meses. Su foco es enseñar cómo construir un mecanismo de búsqueda y una red social. Como señala Ben-Ari [46], el estudiante no es desafiado a construir un programa desde el comienzo, ya que se brindan muchos programas ya prearmados. No incluye prácticas de ingeniería de software en forma explícita. Otro ejemplo es "Informática 101" disponible en Coursera, que enseña los conceptos básicos de la computación e ideas básicas, como abstracción. Utiliza una variante de JavaScript como lenguaje de programación. El curso no profundiza en los aspectos de programación en sí [47].

En relación a MOOCs diseñados para jóvenes podemos citar (en inglés) "MyCS: Ciencias de la computación para principiantes", ofrecido por Harvey Mudd College [48]. Explora cómo funcionan las computadoras y cómo las podemos usar para resolver problemas interesantes. Dura 5 semanas, donde se alternan tópicos generales de ciencias de la computación y actividades de programación en Scratch. Incluye muchas guías de texto y pocos videos, siendo éstos no de alta calidad de producción, según reconocen los propios autores del curso. Estos factores podrían ser desmotivantes para jóvenes. "Programando en Scratch" (en inglés) es otro MOOC también ofrecido por la misma institución [49]. Dura 6 semanas, incluye los conceptos de Scratch pero no cubre temas de pensamiento computacional o ingeniería de software.

En español hay menos disponibilidad. Podemos referir a "Pensamiento Computacional en la Escuela", MOOC de 5 semanas ofrecido en Miríada [50]. Consta de 2 partes: a) introducción conceptual al pensamiento computacional y su aplicación en la vida diaria e b) introducción práctica a Scratch. Aunque el curso promueve una reflexión profunda acerca de los conceptos de pensamiento computacional, el contenido en video puede ser poco atractivo para jóvenes, ya que la mayoría de ellos son presentaciones de imágenes, con voz de fondo. Los ejemplos en Scratch se presentan completos y luego se dividen en sus componentes, a partir de lo cual se reconstruye el programa. Este enfoque, en nuestra opinión, podría no fomentar el proceso típico de resolución de problemas que se trata de desarrollar en los jóvenes. "SM4T: MOOC de Scratch para jóvenes" es un MOOC ofrecido a través de Plan Ceibal para los jóvenes de Uruguay [51]. No está accesible a estudiantes del resto del mundo. Incluye conceptos de programación a través de múltiples ejemplos pero no refiere a prácticas de ingeniería de software.

En resumen, hay múltiples opciones para aprender y enseñar los conceptos básicos de ciencias de la computación y programación. Considerando como puntos relevantes para un MOOC orientado a jóvenes que incluya tópicos interesantes y divertidos, tareas de complejidad acorde al curso, diferentes tipos de evaluación (como por ejemplo, evaluación entre pares), y videos atractivos y bien diseñados, no hay ninguno de los anteriormente presentados que tenga todas estas características.

V. MOOCs: "¡A programar!" y "Code yourself!"

La Universidad ORT Uruguay y The University of Edinburgh diseñaron en forma conjunta un MOOC de 5 semanas de duración dirigido a jóvenes con los objetivos de fomentar el pensamiento computacional, conocer las prácticas básicas de la ingeniería de software y desarrollar programas. La versión en español se denominó "¡A Programar!" [4] y la versión en inglés "Code Yourself!" [5]. El lenguaje utilizado fue Scratch.

El diseño instruccional fue realizado en conjunto entre las dos universidades, tomándose también en cuenta las recomendaciones de diseño para MOOC presentadas. Se definió para cada unidad cuáles serían los temas teóricos a presentar y el conjunto de ejemplos y aplicaciones a utilizar. También se diseñaron todas las preguntas y demás elementos para la evaluación del curso.

El temario del curso detallado por unidades es:

- "Tu primer programa". Incluye la definición de algoritmo, nociones sobre estructuras de control: secuencia, decisión e iteración, desarrollo incremental de programas y prueba simple. Se introducen los principales componentes de Scratch (ej.: movimiento, apariencia).
- "¡Repetir, repetir, repetir!" Aquí se continúa trabajando con las estructuras de control, analizando en detalle distintos tipos de iteración: controlada por cantidad y controlada por condición. Se incluyen nociones sobre descomposición de problemas, programación orientada a eventos, diseño de interfaz, y detección de requerimientos para la construcción de software. De Scratch se incluyen elementos de la paleta de Control, Eventos, Lápiz, Sonido y Sensores, entre otros.
- "Reinventar juegos". A partir de juegos ya existentes, se crean nuevas versiones. Se detallan prácticas de la ingeniería de software para probar y documentar los programas. Se incluye la noción de iteración anidada. Se usan las paletas de Datos y de Operadores en Scratch.
- "Reutilizando tu código". Las nociones de re-uso de código, generalización, modularidad y uso de parámetros se presentan en esta unidad. Se usa la paleta de "Más bloques", se clonan objetos y se presenta la "mochila" en Scratch.
- "Pensando como ingeniero/a de software". A través del desarrollo de un juego completo, se siguen todos los pasos del desarrollo de software: requerimientos, diseño, implementación, prueba. Se presentan también nociones de complejidad computacional y concurrencia. Se brindan sugerencias para continuar en otros

lenguajes. De Scratch se presenta el envío de mensajes y la publicación en su propio sitio [22].

En cada unidad se incluyen múltiples videos cortos descargables (de entre 3 y 5 minutos de duración en general) con subtítulos opcionales. Se incluyen preguntas dentro de los propios videos con el fin de fomentar el involucramiento de los participantes [30]. Además hay ejemplos de código descargable en Scratch asociado a lo presentado en los videos y recursos útiles adicionales, tales como imágenes, sonidos o links. En los ejemplos de código en Scratch se procuró cubrir temas de interés para jóvenes, tanto mujeres como varones. Algunos ejemplos de programas son: representar un baile, hacer una película, crear un karaoke, implementar el juego "Fruit Ninja" (juego en el que se cortan con la mano frutas que aparecen en la pantalla), romper burbujas con las manos y barrer zombis.

En el curso se decidió incluir un personaje animado de nombre "Cody", con forma de extraterrestre (ver Fig. 1), que interactúa con las docentes para formular preguntas o comentarios. El objetivo de su inclusión es mantener el interés del estudiante joven, agregar algún elemento llamativo y sobre todo, brindar un espacio más para la reflexión, ya que las preguntas que formula el personaje son siempre dirigidas a ampliar lo presentado o realizar alguna aclaración.

Fig. 1. Imagen de "Code Yourself!": docente y "Cody"

Además, en cada unidad se incluyen entrevistas en video a personas relacionadas a la computación y de otras áreas. El objetivo es establecer vínculos entre los conceptos presentados en la unidad con la vida práctica. Por ejemplo, en la Unidad 1 se entrevistó a una productora audiovisual en un estudio de televisión, quien a través del relato del proceso de filmación de programas televisivos presentó la noción de secuencia y el uso de las decisiones. Otro ejemplo es en la Unidad 3, donde se entrevistó a una experta en Biotecnología en su laboratorio, quien refirió a sus tareas cotidianas. A partir de esta entrevista, se pueden identificar en otro contexto el concepto de algoritmo y el proceso de refinamiento o "recreación" de un proceso, o sea, ajustarlo y mejorarlo en función de los resultados obtenidos. La mitad de las entrevistas se realizaron en español y la otra mitad en inglés, incluyéndose subtítulos, reforzando el vínculo entre las dos versiones del curso.

Se agregaron foros (así como sugerencias y recomendaciones para su uso) para permitir y fomentar la discusión entre pares. La participación en los foros no es obligatoria. En los videos se incluyeron tópicos a discutir en el foro, por ejemplo, "formular un algoritmo para cepillar los dientes de Cody", "presentar ejemplos de abstracción en cuadros famosos" o "identificar casos de iteración", donde cada estudiante formula una propuesta, el siguiente responde y formula una nueva propuesta. En forma conjunta se decidió disminuir la participación docente en los foros y reservar esas participaciones a aclaraciones urgentes o de funcionamiento. Se trató de favorecer la comunicación entre pares. Para hacer más vívido el curso, semanalmente se realizaron anuncios en el propio sitio del curso y se enviaron correos electrónicos con las novedades relevantes.

Para la evaluación se trató que sus distintos componentes fomentaran el análisis y la reflexión. La evaluación del curso consta de cuestionarios múltiple opción (máximo 50 puntos) y la realización de dos trabajos con evaluación por pares (máximo 50 puntos). Para aprobar se requiere 50% de los puntos. Los cuestionarios se pueden realizar múltiples veces y se toma el último puntaje obtenido. Se brinda retroalimentación no sólo de "respuesta correcta" o "incorrecta", sino que se otorgan "pistas" o sugerencias en caso de responder mal. Estas características (múltiples intentos y dar retroalimentación rápidamente) fomentan el aprendizaje activo [52]. El primer trabajo corresponde a una animación, en la cual dos personajes interactúan y presentan características de un país o ciudad y deben conversar sobre algún aspecto de interés, como por ejemplo, monumentos, deportes o comidas típicas. El segundo es un juego, en el que se debe llevar a Cody en su nave de regreso a su planeta. Para poder mover la nave se usarán las flechas del teclado, habrá meteoritos que hay que evitar chocar, se llevará la cuenta de los choques y del tiempo de juego. También debe haber sonidos y variados fondos para lograr el efecto de animación. Deben incluirse además instrucciones de juego. En ambas tareas, se brindan guías detalladas de qué elementos incluir (personajes, fondos, diálogos, sonido, etc.). Cada estudiante debe subir el programa y luego, la plataforma le asigna 3 tareas de sus pares para evaluar a través de una detallada rúbrica cuantitativa y en la que además debe incluir comentarios cualitativos.

El curso se diseñó en su totalidad en forma conjunta. La versión final en español (videos, cuestionarios, materiales, etc.) fue desarrollada por la Universidad ORT Uruguay y la versión final en inglés por The University of Edinburgh. Para la filmación local de los videos se siguieron las mismas pautas: utilizar fondos difusos que mantengan el interés pero no distraigan la atención, el uso de ropa de manga larga por cuestiones de inclusión cultural, color de vestuario definido para cada unidad como refuerzo visual de la misma, uso de distintos formatos: por ejemplo, docente a la derecha de la pantalla para incluir textos o imágenes a la izquierda (ver Fig. 2), docente en el centro cuando se hacen programas y uso de "transiciones" (breves cortes de imagen) en los videos para marcar puntos de cierre. Se tuvieron en cuenta las recomendaciones para el diseño de los videos presentadas anteriormente y se contó además con el asesoramiento de especialistas en comunicación audiovisual para su filmación y edición.

Fig. 2. Imagen de "¡A Programar!"

En ambos cursos se incluyó además un video conjunto de apertura y cierre con ambas docentes para reforzar más la idea de un curso único en dos versiones. También se agregó un video de "recorrida por el curso", para mostrar rápidamente los distintos elementos que contiene.

Todos los materiales adicionales para los videos también fueron compartidos en ambas implementaciones. Se incluyeron recursos de tipo "stop-motion" (técnica de animación que consiste en aparentar el movimiento de objetos estáticos por medio de una serie de imágenes fijas sucesivas), gráficos, dibujos, sonidos y el uso de placas con definiciones y algoritmos.

Para la creación y desarrollo de estos cursos se trabajó a distancia, utilizando herramientas de trabajo colaborativo como Google Docs y se realizó una instancia presencial en Edimburgo para fortalecer vínculos y ajustar enfoques. Además de las docentes a cargo de los materiales, el equipo de producción incluyó diseñadores gráficos, expertos en educación, productores audiovisuales, editores, maquilladores, expertos en plataformas y coordinadores locales, además de los especialistas en comunicación ya citados.

VI. Dictados y Modalidades

"¡A Programar!" y "Code Yourself!" están disponibles en la plataforma Coursera desde marzo de 2015. El curso fue anunciado por múltiples medios: a través de redes sociales, instituciones educativas y la propia página de Coursera. La primera edición fue entre marzo y abril de 2015. A partir agosto de 2015, debido a cambios en la plataforma de Coursera, se lanzó en formato "de cohorte" o sesión "flexible", con comienzos mensuales y posibilidad de cambio entre sesiones.

Se detallarán las características de las distintas modalidades del curso según las ediciones ofrecidas en la plataforma Coursera. Estas características incluyen, como se indicó, aspectos vinculados a fecha de apertura y cierre de la sesión, diseño de la interfaz de Coursera, evaluación y fechas de entrega, y certificación. También hubo diferencias en las sesiones en cuanto a participación docente y uso de los foros.

A. Primera edición: marzo/abril 2015

La primera sesión del curso fue entre marzo y abril de 2015, en formato de sesión "fija", esto es, una única sesión con fechas preestablecidas y fijas. En esta primera edición del curso, Coursera ofrecía ciertas características particulares que detallaremos. La interfaz lucía muy similar en ambos idiomas, como se muestra en la Fig. 3 y Fig. 4.

En la Fig. 3 se observa que el curso se estructura en "Anuncios", "Lecciones en video", "Recursos" y "Foros". Dentro de "Actividades", se encuentran los cuestionarios, evaluaciones por pares y encuestas. La información general del curso está en "Acerca del curso", donde se detalla el temario, formato y evaluación así como el equipo docente y de producción.

Fig. 3. Sitio de la primera sesión de "¡A Programar!"

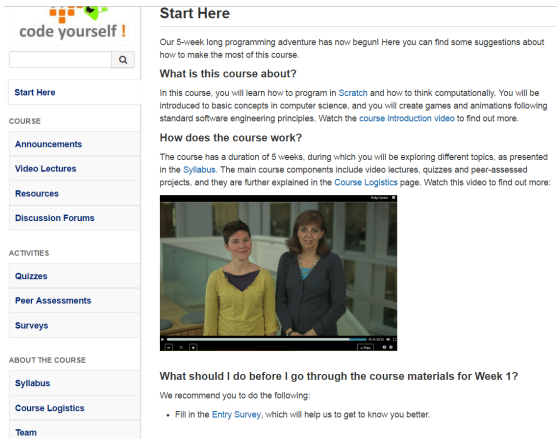

Fig. 4. Sitio de la primera sesión de "Code Yourself!"

En la Fig. 5 se observa el detalle de los videos por unidad en "¡A Programar!" (es similar en "Code Yourself!"). Cada video incluye su título, duración, y los recursos que dispone: foro/s específico/s, ejemplos de código asociado, subtítulos y la opción de descarga.

Se habilitaron las unidades de a una por semana. Las fechas de entrega de cuestionarios y trabajos estaban preestablecidas y eran fijas (no se permitía entrega tardía). La evaluación constaba de los cuestionarios y trabajos detallados, pero ninguno era obligatorio. Se requería en total 50% de los puntos. Por ejemplo, un estudiante podría aprobar el curso solamente respondiendo correctamente todos los cuestionarios. La evaluación por pares era anónima: los estudiantes no conocían la identidad de quién los corregía ni de a quiénes corregían. Al cumplir los requisitos de aprobación, Coursera

emitía gratuitamente una constancia de compleción en formato pdf. Si se obtenía 90% o más de los puntos, la constancia indicaba además que se había realizado con "distinción". Esta constancia gratuita no verificaba identidad del participante, ofreciéndose también la opción de obtener una constancia paga con verificación de identidad.

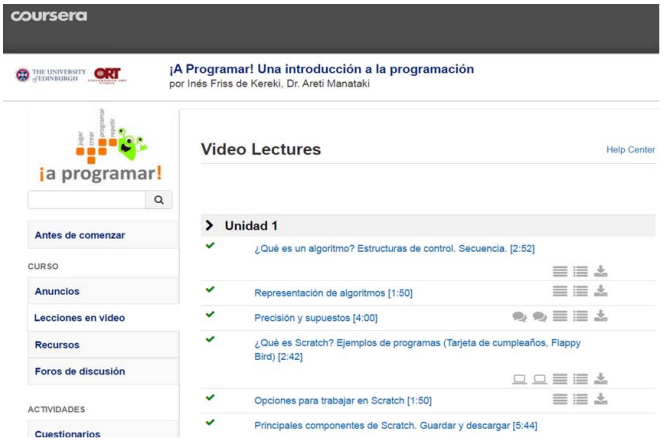

Fig. 5. Detalle de los videos y demás materiales por unidad en "¡A Programar!"

En la primera sesión hubo participación de los docentes a cargo del curso así como de los docentes asistentes a través de los anuncios semanales en el propio sitio y por correo electrónico. Al comienzo de cada semana se publicó un anuncio con lo que se vería esa semana, detallando los temas a tratar. También se incluyeron novedades específicas, por ejemplo, cerca de la fecha de entrega de una tarea el sitio de Scratch iba a estar "no disponible" debido a mantenimiento no previsto inicialmente; se informó sobre esta situación y se ajustaron las fechas de entrega de esa tarea. En los foros, la presencia docente se mantuvo baja, como se acordó.

B. Siguientes ediciones: desde agosto de 2015

De la primera sesión se encontraron oportunidades de mejora a través de múltiples reuniones virtuales entre los integrantes de los dos equipos. En ellas, se decidió mantener el enfoque y los contenidos, haciendo leves ajustes en las evaluaciones, modificándose los puntajes requeridos de evaluación, estableciéndose 50% para cada cuestionario y trabajo. También, se decidieron incluir nuevos recursos, como documentos con información sobre cómo subir trabajos y descargar archivos, preguntas que aparecieron muy frecuentemente durante la primera edición. Algunas preguntas y respuestas de los cuestionarios fueron ajustadas, en función de dudas que habían surgido.

En las sesiones a partir de agosto de 2015 hubo cambios en cuanto a la inscripción, debido a modificaciones en Coursera. En la nueva modalidad, el estudiante se inscribe para la sesión actual (comienzo mensual). Se habilita un período de algunos días para esta inscripción. Por ejemplo, si el curso inicia el 25 de abril y termina el 5 de junio, se permite inscribir hasta el 5 de mayo. Al inscribirse en forma temprana (previo al comienzo del curso), se habilita la visualización de la primera semana del curso de forma de motivar a comenzar lo antes posible, sin necesidad de esperar al comienzo efectivo. Se habilita también el primer cuestionario. Luego del comienzo efectivo del curso se habilitan todas las unidades.

Debido a la multiplicidad de sesiones, no se incluyó el anuncio semanal y se agregó un texto introductorio a cada semana en el propio sitio.

La interfaz de la plataforma Coursera se ha rediseñado y simplificado, según se observa en la Fig. 6. Los contenidos está organizados por semanas (de la misma forma que antes se denominaban "unidades")

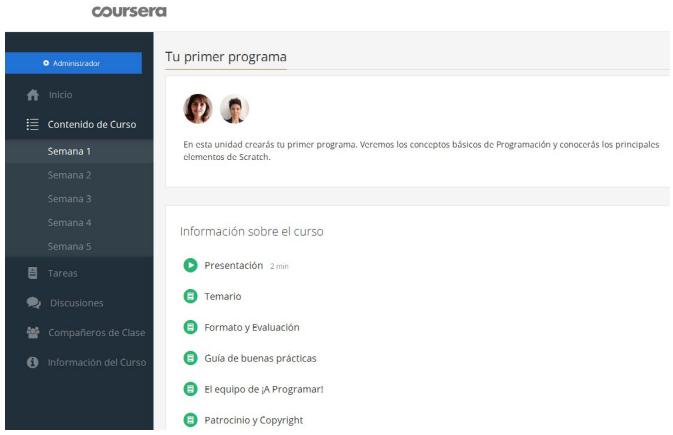

Fig. 6. Sitio de las nuevas sesiones de "¡A Programar!"

Como novedad, en cada sesión, Coursera indica la cantidad de compañeros de clase de la sesión actual y su distribución geográfica (sin individualización particular), con el fin de generar sentido de comunidad y pertenencia (Fig. 7).

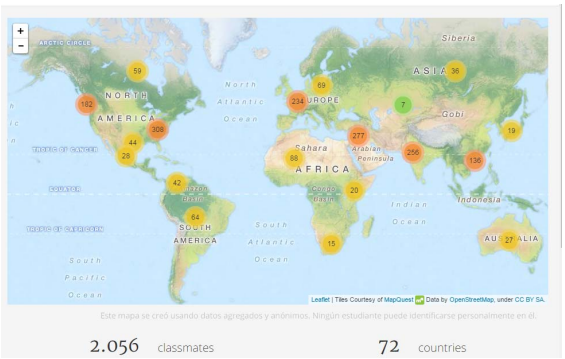

Fig. 7. Compañeros de clase y países (imagen de "Code Yourself!")

Como se indicó, se ajustó la evaluación para asegurarse realizar cada actividad y tarea propuesta, otorgando además flexibilidad en cuanto a las fechas de entrega. En caso de haber llegado a la fecha de terminación del curso y no haber completado las actividades, se brinda la oportunidad de cambiar a la siguiente sesión, manteniendo lo realizado. Cada cuestionario se puede realizar hasta 5 veces por día. Como se refirió, la evaluación por pares en la primera sesión era anónima. En las ediciones a partir de agosto se incluye en la plataforma la información sobre el autor así como de los correctores. También se brinda información sobre el puntaje particular otorgado por cada evaluador (en la primera edición sólo se incluía el promedio otorgado). Además es posible,

luego de tener las evaluaciones de los pares, revisar la tarea y volver a subirla, con el objetivo de mejorar los propios resultados. A excepción de alguna consulta puntual en los foros, no hubo reclamos sobre la evaluación brindada por los pares.

Un cambio importante que incluyó Coursera fue que no se emite más la constancia de completación. En la cuenta personal de cada estudiante aparece que se completó el curso, pero no hay certificado específico como en la primera sesión. Se mantiene el certificado pago, con verificación de identidad a través de cámara web y patrón de escritura.

Coursera incluyó en los cursos el rol de "Mentor Comunitario", que son voluntarios que ayudan con la atención de los foros. De la primera sesión se seleccionaron aquellos participantes que más y mejor participaron en los foros y se los invitó a colaborar en las nuevas sesiones. La respuesta a la invitación fue positiva. En las primeras sesiones del nuevo formato colaboraron activamente.

VII. Resultados

En esta sección incluimos resultados de las dos modalidades. La primera modalidad corresponde a la sesión de marzo-abril 2015. La segunda modalidad incluye los cambios presentados (esto es: ajustes de puntajes y de algunas preguntas de cuestionarios, incremento de recursos, modificaciones en Coursera acerca de interfaz, inscripción, cambio de sesión y fechas de entrega, ausencia de certificado de completación, repetición de cuestionarios, evaluación por pares no anónima y "mentor"). Para esta segunda modalidad, se presenta el resultado conjunto de las sesiones desde agosto 2015 hasta abril 2016. Detallaremos resultados generales, demografía, distribución geográfica de los participantes y evaluación del curso para ambas modalidades respectivamente.

A. Primera sesión

Según se presenta en la Tabla I, más de 59.500 personas se inscribieron en "Code Yourself!" y más de 25.200 lo hicieron en "¡A Programar!". La encuesta inicial (no obligatoria) fue respondida por aproximadamente un 10% en la de inglés y 25% en la de español. En relación al conocimiento previo de programación, más del 82% de participantes indicaron no saber o saber muy poco.

El porcentaje de participantes que completó el curso fue 2.68% y 6.30% respectivamente. La retención se puede definir como el porcentaje de alumnos que se inscriben y cumplen los requisitos de aprobación establecidos para el curso [53]. La alta deserción es la norma en los MOOCs [54]. En los cursos de Coursera es de 5% aproximadamente [53]. Ho et al refieren 7% de tasa de completación en ciencias de la computación [55].

La retención debe ser considerada en el contexto de la intención del estudiante, variedad de conocimientos previos y motivaciones [53]. Muchos participantes de los MOOCs al inscribirse no tienen la intención de finalizar el curso. Como indica Zheng et al [33], hay variedad de razones para inscribirse a un MOOC, completar el curso es sólo una. Algunas personas, por ejemplo, se inscriben para acercarse a un área particular de estudio, pero no sienten la necesidad de ver todas las lecciones o completar todas las tareas asignadas [33]. Es interesante destacar que, según manifestaron los participantes en la encuesta inicial, las principales motivaciones en el curso fueron "Aprender cosas nuevas" (referido por más del 92% de los encuestados, en ambos cursos) y "mejorar las opciones de carrera (indicado por más del 46%). La intención de aprobar el curso fue de 27% en "Code Yourself!" y 37% en "¡A Programar!".

TABLA I. Resultados de la primera sesión

	1era sesión	
	Code Yourself!	***¡A Programar!***
Inscriptos	59531	25255
Completaron encuesta inicial	6229 (10.46%)	6429 (25.45%)
¿Sabía de programación antes de comenzar?	82% no	89% no
Completaron satisfactoriamente el curso	1595	1592
% completaron/inscriptos	2.68%	6.30%

En la versión en inglés hubo distribución casi pareja por género, mientras que en la versión en español la relación fue 2 hombres por cada mujer (ver Tabla II). Esto implica alta participación de ambos géneros. Ho et al [55], refieren que, en promedio en cursos de ciencias de la computación, la relación es de cuatro hombres por cada mujer. El porcentaje de jóvenes en esta primera sesión es algo superior al presentado por Nesterko et al [56], quienes refieren 7.8% (incluyendo hasta 20 años) y al indicado por Basogain et al [57], de 2.85% (en menores de 18 años). Es de destacar que en "¡A Programar!" prácticamente todos los participantes tenían como lengua materna la del curso, hecho que no sucede en "Code Yourself!", donde la mitad no la tiene. Quizás esta diferencia sea la justificación de por qué se tuvo mayor proporción de encuestas respondidas en "¡A Programar!", suposición que debe verificarse con posteriores estudios.

En "Code Yourself!" hubo amplia representación de participantes de América del Norte y Central, Asia y Europa. Los principales países fueron Estados Unidos seguido de India. En el caso de "¡A Programar!", tuvo mayor representación América del Sur, seguido por América del Norte y Central. Los países con mayor participación fueron México y España (Tabla III).

TABLA II. Demografía de la primera sesión

	1era sesión	
	Code Yourself!	***¡A Programar!***
Género	54% Hombres, 44% Mujeres, 2% no responde	65% Hombres, 34% Mujeres, 1% no responde
Jóvenes (< = 18 años)	9%	15%
¿Misma lengua del curso?	51% sí	97% sí

TABLA III. DISTRIBUCIÓN GEOGRÁFICA DE LA PRIMERA SESIÓN

	1era sesión	
	Code Yourself!	***¡A Programar!***
Países	197	117
Continentes	América del Norte y América Central: 37%, Asia: 28%, Europa: 25%, Africa: 5%, América del Sur: 4%, Resto del mundo: 1%	América del Sur: 40%, América del Norte y América Central: 35%, Europa: 23%, Asia:1,6%, Resto del mundo: 0,4%
Primeros 5 países	Estados Unidos: 31%, India: 13%, Reino Unido: 4%, China: 4%, Canadá: 3%	México: 20%, España: 20%, Colombia: 12%, Estados Unidos: 7%, Uruguay: 6%

El curso en ambas lenguas fue valorado positivamente desde varios puntos de vista: duración adecuada (73% o más así lo indicaron), ritmo correcto (>=79%), cubrió o superó las expectativas (>= 93%) y lo recomendarían (>=96%). Los elementos más valiosos fueron los videos, seguidos en cada curso por cuestionarios y evaluación por pares (Tabla IV). A futuro, 90% o más de los estudiantes planifica seguir aprendiendo programación.

TABLA IV. SOBRE EL CURSO (1ERA SESIÓN, ENCUESTA FINAL)

	1era sesión	
	Code Yourself! (896 encuestas)	***¡A Programar! (1587 encuestas)***
Duración	73% adecuada 21% algo corta 3% algo larga 3% muy larga/muy corta	78% adecuada 18% algo corta 2% algo larga 2% muy larga/muy corta
Ritmo del curso	81% correcto	79% correcto
¿Cubrió/Superó las expectativas?	93% sí	95% sí
¿Recomendaría el curso?	96% sí/probablemente	98% sí/probablemente
Elementos más valiosos del curso (se pueden elegir varios elementos)	93% video 54% cuestionarios 53% evaluación por pares 42% recursos 19% foros 11% interacción	96% video 65% cuestionarios 60% evaluación por pares 53% recursos 27% foros 18% interacción
¿Planea seguir aprendiendo programación?	95%	90%

B. Sesiones posteriores

La cantidad total de inscriptos en esta modalidad es algo inferior a la de la primera sesión (Tabla V). Se observa además porcentajes más bajos de respuesta en relación a la encuesta inicial (aproximadamente 3% en "Code Yourself!" y 11% en "¡A Programar!"). Una posible explicación preliminar podría ser la ausencia de los anuncios semanales, a través de los cuales en la primera sesión se hizo especial énfasis en contestar la encuesta. En relación al conocimiento previo de programación, 72% de participantes del curso en inglés y 83% en español indicaron no saber o saber muy poco. Esto implica que para la gran mayoría de los participantes el curso representaba un área de conocimiento nuevo, al igual que en la primera sesión.

Si bien las motivaciones para realizar el curso fueron similares, la intención de aprobar el curso fue de 20% en "Code Yourself!" y 31% en "¡A Programar!", valores algo más bajos que en la primera sesión. Los porcentajes de compleción fueron similares a la primera sesión en inglés y algo menores en español, pero cercano a los rangos de compleción de Coursera presentados.

TABLA V. RESULTADOS ACUMULADOS DE LAS SIGUIENTES SESIONES

	Siguientes sesiones	
	Code Yourself!	***¡A Programar!***
Inscriptos	30330	23965
Completaron encuesta inicial	895 (2.95%)	2735 (11.41%)
¿Sabía de programación antes de comenzar?	72% no	83% no
Completaron satisfactoriamente el curso	809	739
% completaron/inscriptos	2.67%	3.08%

La distribución por género se mantuvo en "¡A Programar!" y en "Code Yourself!" hubo una participación mayor de mujeres que de hombres (ver Tabla VI). La proporción de jóvenes fue más baja. Se sigue observando que en la versión en inglés cerca de la mitad de los participantes tienen otra lengua materna. En la de español la amplia mayoría tiene como primera lengua la del curso.

TABLA VI. DEMOGRAFÍA DE LAS SIGUIENTES SESIONES

	Siguientes sesiones	
	Code Yourself!	***¡A Programar!***
Género	42% Hombres, 56% Mujeres, 2% no responde	66% Hombres, 33% Mujeres, 1% no responde
Jóvenes (< = 18 años)	6%	5%
¿Misma lengua del curso?	42% sí	97% sí

La distribución geográfica de participantes por continentes y países (ver Tabla VII) se mantuvo bastante similar a la presentada para la primera sesión.

La duración del curso fue percibida mayoritariamente también como adecuada (Tabla VIII). El ritmo del curso fue valorado mejor: de 79-81% en la primera sesión (Tabla IV) a 90-91% (Tabla VIII). Algunos posibles factores, sujetos a posterior análisis para confirmar o no su influencia, pueden ser los ajustes en la modalidad: flexibilidad en cuanto a fechas de entrega, posibilidad de cambio de sesión y disponibilidad de todo el material desde el comienzo y aún previo al mismo.

Si bien el curso en la sesión inicial tanto en español como en inglés ya había sido valorado positivamente, en estas nuevas sesiones se mejoró aún más dicha valoración, superando el porcentaje de 98.6 tanto en cuanto a cubrir/superar las expectativas como en relación a recomendarlo (ver Tabla VIII).

TABLA VII. DISTRIBUCIÓN GEOGRÁFICA DE LAS SIGUIENTES SESIONES

	Siguientes sesiones	
	Code Yourself!	***¡A Programar!***
Países	188	107
Continentes	América del Norte y América Central: 35%, Asia: 28%, Europa: 25%, Africa: 6%, América del Sur: 4%, Resto del mundo: 2%	América del Sur: 43%, América del Norte y América Central: 37%, Europa: 18%, Asia:1%, Resto del mundo: 1%
Primeros 5 países	Estados Unidos: 28%, India: 13%, Reino Unido: 5%, Canadá: 4%, China: 3%	México: 28%, España: 16%, Colombia: 13%, Perú: 8%, Argentina: 6%

En relación a los recursos (links, archivos, etc.) según se observa en la Tabla VIII, mejoró la opinión. En la sesión inicial 42%-53% de quienes respondieron lo señalaron como valioso (Tabla IV). En estas nuevas ediciones fue valorado como positivo en 65% o más de los casos. Los cuestionarios y la evaluación por pares también tuvieron mejora en sus porcentajes como elementos valiosos. La explicación podría ser el ajuste de los cuestionarios, de la evaluación por pares así como los recursos agregados.

TABLA VIII. SOBRE EL CURSO (SIGUIENTES SESIONES, ENCUESTA FINAL)

	Siguientes sesiones	
	Code Yourself! (146 encuestas)	***¡A Programar! (222 encuestas)***
Duración	79% adecuada 18% algo corta 2% algo larga 1% muy larga/ muy corta	89% adecuada 8% algo corta 2% algo larga 1% muy larga/ muy corta
Ritmo del curso	90% correcto	91% correcto
¿Cubrió/superó las expectativas?	98,7% sí	98,6% sí
¿Recomendaría el curso?	98.6 % sí/probablemente	98.6% sí/probablemente
Elementos más valiosos del curso (se pueden indicar varios)	99% video 69% cuestionarios 62% evaluación por pares 65% recursos 16% foros 12% interacción	98% video 72% cuestionarios 68% evaluación por pares 67 % recursos 22% foros 20% interacción
¿Planea seguir aprendiendo programación?	97.2% sí	94.6% sí

En cuanto a los foros y la interacción, se tienen porcentajes algo inferiores a los de la primera sesión, por lo que podría inferirse que el rol de mentor no tuvo un impacto significativo positivo en la percepción de valía de esos elementos. La intención de seguir aprendiendo a programar es alta: más del 94% así lo indican.

C. Resultados generales

En términos generales, en cualquiera de sus modalidades, el curso tuvo gran aceptación y cumplió las expectativas de los participantes, según sus respuestas a las encuestas. Las mejoras en cuanto a recursos y ajustes en la evaluación tuvieron impacto positivo en la opinión de los participantes. Los ajustes en cuanto a flexibilidad podrían explicar la mejora de la opinión en cuanto al ritmo.

Es importante destacar que las condiciones iniciales de los participantes son similares: la gran mayoría manifestó no saber previamente de programación. Además, 90% o más indican que planea seguir aprendiendo programación. Evaluando todos los resultados, parecen indicar que los objetivos planteados del curso se han logrado.

VIII. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se presentó la experiencia realizada por la Universidad ORT Uruguay y The University of Edinburgh en cuanto a la realización conjunta y seguimiento de un MOOC en dos idiomas disponible en Coursera, dirigido principalmente a jóvenes con el objetivo de fomentar el pensamiento computacional, aprender a programar y conocer las prácticas básicas de la ingeniería de software.

El curso se dictó simultáneamente en una sesión inicial "fija" y luego, con ajustes, se cambió de modalidad a "flexible". Estos cambios incluyen, entre otros, comienzos simultáneos mensuales, posibilidad de cambio de sesión y disponibilidad del material en su totalidad desde el comienzo. En ambas modalidades e idiomas se observan resultados similares: alto grado de satisfacción con el curso y marcada expresión de interés en seguir aprendiendo a programar, partiendo la gran mayoría de manifestar no tener conocimientos previos de programación. Los cambios menores introducidos mejoraron la percepción del curso en cuanto a recursos y ritmo.

Como actividades futuras, se planea profundizar en el estudio de las diferencias entre modalidades y su impacto. También se está trabajando con la Universidad de San Pablo para lanzar una versión en portugués del curso ("Programe-se"), produciéndose los videos directamente en ese idioma.

REFERENCIAS

[1] B. Trilling y C. Fadel, "21st century Skills, learning for life in our time", Jossey-Bass, San Francisco, 2009

[2] ACM Computer Science Teachers Association. CSTA K–12 Computer Science Standards (2011) http://csta.acm.org/Curriculum/sub/K12Standards.html. Accedido Diciembre 15, 2015.

[3] A. Balanskat, y K. Engelhardt. (contributors), "Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europa". European Schoolnet, Belgium. October 2015.

[4] MOOC: "¡A Programar!". Universidad ORT Uruguay and The University of Edinburgh. https://www.coursera.org/learn/a-programar. Accedido Diciembre 15, 2015.

[5] MOOC: "Code Yourself!". The University of Edinburgh and Universidad ORT Uruguay. https://www.coursera.org/learn/intro-programming. Accedido Diciembre 15, 2015.

[6] Framework for 21st Century Learning, http://www.p21.org/storage/documents/1.__p21_framework_2-pager.pdf Accedido Abril 26, 2016.

[7] K. Falkner, R. Vivian, y N. Falkner, "Teaching Computational Thinking in K-6: The CSER Digital Technologies MOOC", ACE 2015, Sydney, Australia, 2015.

[8] I. Lee, F. Martin y K. Apone (2011). "Integrating Computational Thinking Across the K-8 Curriculum". ACM Inroads, Volume 5 Issue 4, p. 64-71, Diciembre 2014

[9] J. Wing, "Computational thinking". Communications of the ACM, Vol. 49, No. 3, March 2006.

[10] CSTA: Operational Definition of Computational Thinking for K-12 Education, https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf Accedido Marzo 14, 2016.

[11] A. Bundy, "Computational thinking is pervasive". J. Sci. Pract. Comput. 1, 67–69, 2007.

[12] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., y J. T. Korb, "Computational thinking in elementary and secondary teacher education". ACM Trans. Comput. Educ.14, 1, Article 5, March 2014.

[13] Mitchel Resnick, y David Siegel, "A Different Approach to Coding: How kids are making and remaking themselves from Scratch" https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.524r17diw. Accedido Abril 23, 2016.

[14] M. Sharples, A. Adams, N. Alozie, R. Ferguson, E, FitzGerald, M. Gaved, P. McAndrew, B. Means, J. Remold, B. Rienties, J. Roschelle, K. Vogt, D. Whitelock, y L. Yarnall, "Innovating Pedagogy 2015", Open University Innovation Report 4, Milton Keynes: The Open University, United Kingdom, 2015

[15] S. Grover, y R. Pea, "Computational Thinking in K–12. A Review of the State of the Field", Educational Researcher, vol. 42 no. 1 38-43, Jan/Feb 2013.

[16] M. Zapata-Ros. "Pensamiento computacional: Una nueva alfabetización digital". RED: Revista de Educación a Distancia, 46(4), Setiembre 2015, DOI 10.6018/red/46/4

[17] M. Armoni, "Early Education – what does computing has to do with it and in what ways?", WiPCSE'15 (Proc of the Workshop in Primary and Secondary Computing Education), ACM, USA, 2015.

[18] L. Mannila, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo, L. Rolandsson, y A. Settle, "Computational Thinking in K-9 Education", ITiCSE-WGR'14, Uppsala, Sweden, 2014.

[19] Department for Education, "National curriculum in England: computing programmes of study (2013)". https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study. Accedido Diciembre 15, 2015.

[20] Code.org, https://code.org/ Accedido Abril 21, 2016.

[21] Codecademy, http://www.codecademy.com/ Accedido Abril 21, 2016

[22] Scratch, http://wiki.scratch.mit.edu/wiki/Scratch Accedido Diciembre 15, 2015.

[23] Alice, http://www.alice.org. Accedido Enero 4, 2016.

[24] MIT AppInventor, http://appinventor.mit.edu/. Accedido Diciembre 15, 2015.

[25] S. Esper, S. Foster, W. Griswold, C. Herrera, y W. Snyder, "CodeSpells: Bridging Educational Language Features with Industry-standard Languages", Koli Calling 2014, Koli, Finland, 2014

[26] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E.Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, y Y. Kafai. "Scratch: Programming for All. Communications of the ACM", Vol. 52 No. 11, Pages 60-67, 2009.

[27] K. Brennan, y M. Resnick, "New Frameworks for Studying and Assessing the Development of Computational Thinking", Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, 2012.

[28] D. Malan, "Implementing a massive open online course (MOOC)", Journal of Computing Sciences in Colleges table of contents archive, Volume 28 Issue 6, p. 136-137, June 2013.

[29] Oxford Dictionary, http://www.oxforddictionaries.com/us/definition/american_english/MOOC. Accedido Abril 22, 2016

[30] W. Siever, "Leveraging MOOCS", JCSC 30, 2, Dec 2014.

[31] Universities UK: "Massive open online courses. Higher education's digital moment?" http://www.universitiesuk.ac.uk/highereducation/Documents/2013/MassiveOpenOnlineCourses.pdf. Accedido Diciembre 16, 2015.

[32] A. Mc Auley, B. Stewart, G. Siemens, y D. Cormier, "The MOOC Model for Digital Practice". http://davecormier.com/edblog/wp-content/uploads/MOOC_Final.pdf Accedido Diciembre 16, 2015.

[33] S. Zheng, M. Rosson, P. Shih, y J. Carroll, "Understanding student motivation, behaviors, and perceptions in MOOCs", CSCW 2015, pp. 1882-1895, Canada, 2015.

[34] S. Cooper, y M. Sahami, "Reflections on Standford's MOOCs", Communications of ACM, V 56, N 2, Febrero 2013.

[35] K. Button, "10 Lessons learned from Moocs", Education Dive, http://www.educationdive.com/news/10-lessons-learned-from-moocs/306113/. Accedido Enero 4, 2016.

[36] C. Alario Hoyos, M. Pérez-Sanagustín, C. Delgado, y P. J. Muñoz-Merino, "Recommendations for the design and deployment of MOOCs: insights about the MOOC digital education of the future deployed in MiríadaX", TEEM '14 Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality, Pp 403-408, ACM New York, NY, USA, 2014.

[37] P. J. Guo, J. Kim, y R. Rubin, "How video production affects student engagement: an empirical study of MOOC videos", L@S '14 Proceedings of the first ACM conference on Learning @ scale conference, USA, 2014.

[38] P. Adamopoulus, "What makes a great MOOC? An interdisciplinary analysis of student retention in online courses". 34th International Conference on Information Systems, Milan, 2013.

[39] Coursera, https://www.coursera.org/. Accedido Diciembre 15, 2015.

[40] edX, https://www.edx.org/. Accedido Diciembre 16, 2015.

[41] Udacity, https://www.udacity.com/. Accedido Marzo 20, 2016

[42] Miríada, https://miriadax.net/home. Accedido Marzo 21, 2016.

[43] Edsurge, "Udacity, Coursera and edX Now Claim Over 24 Million Students", https://www.edsurge.com/news/2015-09-08-udacity-coursera-and-edx-now-claim-over-24-million-students Accedido Abril 28, 2016.

[44] MOOC: "C50X: Introduction to Computer Science", Harvard University https://www.edx.org/course/introduction-computer-science-harvardx-cs50x. Accedido Enero 4, 2016.

[45] MOOC: "Intro to Computer Science". https://www.udacity.com/course/intro-to-computer-science--cs101. Accedido Enero 4, 2016.

[46] M. Ben-Ari, "MOOCs on introductory programming: a travelogue". Acm Inroads, 4(2), 58-61, 2013.

[47] MOOC: "Computer Science 101", https://www.coursera.org/course/cs101, Accedido Abril 15, 2016

[48] MOOC: "My CS: Computer Science for Beginners". https://www.edx.org/course/mycs-computer-science-beginners-harveymuddx-cs001x# Accedido Diciembre 15, 2015.

[49] MOOC: "Programming in Scratch". Harvey-Mudd College. https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-0. Accedido Diciembre 15, 2015.

[50] MOOC: "Pensamiento Computacional en la escuela (2da. edición)" https://miriadax.net/web/pensamiento-computacional-en-la-escuela-2ed. Accedido Diciembre 15, 2015.

[51] I. Kereki, y J. V. Paulós, "SM4T: Scratch MOOC for Teens - A pioneer pilot experience in Uruguay", FIE 2014 Frontiers in Education, Spain, 2014.

[52] M. Bali, “MOOC Pedagogy: Gleaning Good Practice from existing MOOCs”, MERLOT J. of online Learning and Teaching, V. 10, N.1, March 2014.

[53] D. Koller, A. Ng, C. Do y Z. Chen, “Retention and intention in massive open online courses: In depth”. Educause Review, http://er.educause.edu/articles/2013/6/retention-and-intention-in-massive-open-online-courses-in-depth, 2013.

[54] D. Coetzee, A. Fox, M. Hearst, y B. Hartmann., “Should your MOOC Forum use a reputation System?”, CSCW 2014, p. 1176-1187, 2014

[55] A. Ho, I. Chuang, J. Reich, C. Austun Coleman, J. Whitehill, C. Northcutt, J. Williams, J. Hansen, G. Lopez, y R. Petersen, “HarvardX and MITx: Two Years of Open Online Courses Fall 2012-Summer 2014”. Available at SSRN: http://ssrn.com/abstract=2586847 or http://dx.doi.org/10.2139/ssrn.2586847 Accedido Marzo 21, 2016

[56] S. Nesterko, D. Seaton, K. Kashin, Q. Han, J. Reich, J. Waldo, I. Chuang I., y A. Ho, Age Composition (HarvardX Insights, 2014, http://harvardx.harvard.edu/harvardx-insights/age-composition. Accedido Marzo 22, 2016.

[57] X. Basogain, M. Olabe y J. Olave, “Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje”, RED- Revista de Educación a Distancia, 46(6), DOI 10.6018/red/46/6, 2015. Accedido Abril 26, 2016.